Universidade Federal do Paraná – UFPR
Departamento de Engenharia Química

Breve Introdução à Programação em Scilab 6.0

Prof. Éliton Fontana

2018/2

Conteúdo

1 Conceitos Básicos **4**

2 Operando Vetores e Matrizes **6**

2.1 Criando Matrizes de Valores Reais 6

2.2 Acessando e Modificando Elementos de uma Matriz 6

2.3 O Operador ‘ : ’ 7

3 Loops e Estruturas Lógicas **9**

3.1 For 9

3.2 While 10

3.3 If 10

4 Criando e Utilizando Funções **11**

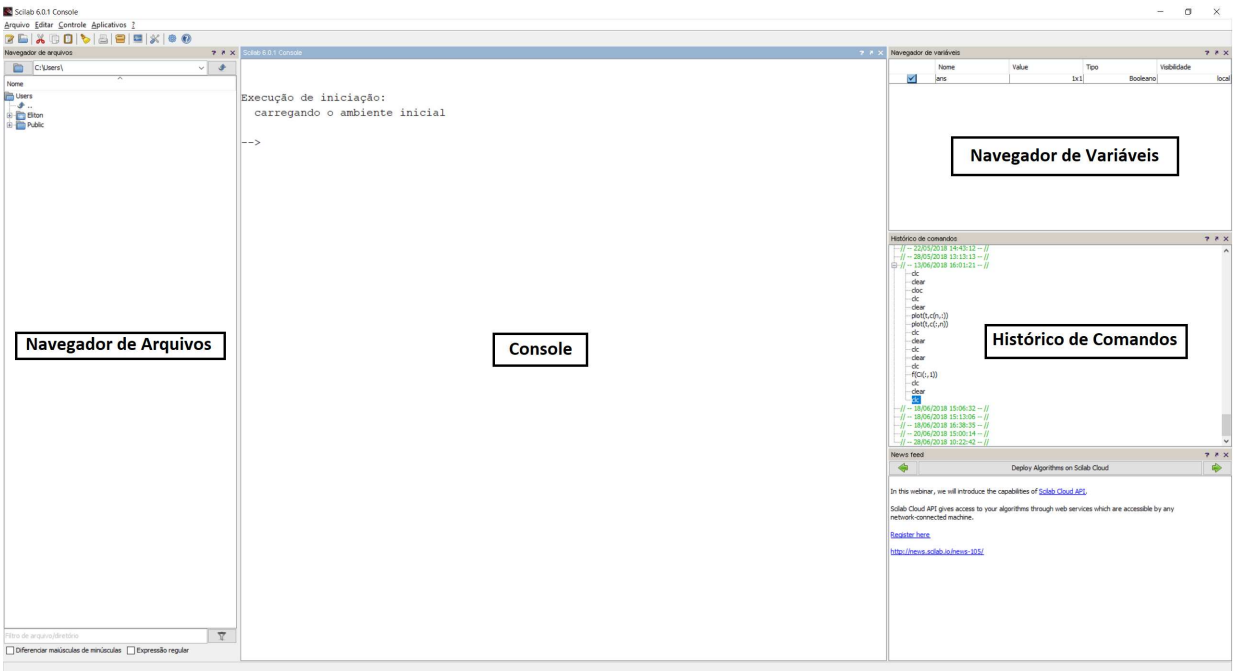
4.1 Funções Definidas no Scilab 14

5 Exercícios Propostos **16**

1 Conceitos Básicos

O Scilab é um software gratuito, de código aberto e multi-plataforma orientado à computação científica. A sintaxe de programação utilizada pelo Scilab é muito similar ao MATLAB, sendo muito fácil a transição entre os dois programas. Assim como MATLAB, o Scilab permite a manipulação de matrizes de maneira simples e direta, possibilitando a resolução de problemas complexos através da aplicação de métodos numéricos.

Após a instalação do programa, a tela inicial exibida será similar à apresentada na figura a seguir.



A tela inicial contém os seguintes elementos:

- ***Navegador de Arquivos:*** Permite navegar pelos arquivos salvos;
- ***Console:*** O console é o espaço onde os comandos podem ser executados. Basta digitar um comando específico e executar com a tecla Enter. Por exemplo, o comando $a = 2$

associa o valor 2 à variável a[1].

- ***Navegador de Variáveis:*** Este espaço permite verificar quais variáveis estão atualmente salvas na memória, bem como a forma destas variáveis e os valores alocados.

- ***Histórico de Comandos:*** Lista os últimos comandos usados no console.

Apesar de o console permitir a definição de variáveis e operações entre eles, ele não é um ambiente adequado para a construção de códigos mais complexos. Para isso, são utilizados os ***scripts***, que são basicamente uma sequência de comandos que será posteriormente executada. O editor de scripts do Scilab é chamado de SciNotes. Para abrir o Scinotes, pode-se clicar em Aplicativos → SciNotes. Para executar um script, basta clicar em Executar → Salva e Executar, ou pressionar F5.

Exercício 01: Utilizando o SciNotes, defina duas variáveis $a = 1$ e $b = 2$. Defina uma nova variável $c = a+b$ e mande imprimir o valor de c no console, através do comando disp(c). Verifique se o valor impresso está correto.

Dica de Sobrevivência 1: É altamente recomendável que todo script comece com os comandos *clear* e *clc*. O comando *clear* limpa as variáveis alocadas na memória, enquanto que o *clc* limpa o console.

Dica de Sobrevivência 2: Para inserir comentários, utiliza-se o comando // no início da linha. Um bom código é um código bem comentado.

[1] O comando 'disp' pode ser usado para imprimir no console o valor de alguma variável ou algum texto específico. Faça um teste, digite *disp*('Hello World!!').

2 Operando Vetores e Matrizes

A principal vantagem da utilização de programas como o Scilab é a facilidade em operar com matrizes. As matrizes são categorizadas com base no número de linhas, número de colunas e tipo de dado (por exemplo, valores reais, inteiros, booleanos, etc.). Do ponto de vista do programa, vetores são interpretados como matrizes com número de linhas ou número de colunas igual a 1. Até mesmo os escalares são interpretados como matriz com dimensão 1×1.

2.1 Criando Matrizes de Valores Reais

Quando se deseja agrupar um determinado conjunto de valores conhecidos em uma matriz, a maneira mais simples é definir os valores entre colchetes simples **'[ ]'**. Elementos em uma mesma linha são separados por vírgula **','**, enquanto que elementos em diferentes colunas são separados por ponto e vírgula **';'**. É importante destacar que ',' é usada para separar argumentos e ***nunca para separar casas decimais***. Para isso, utiliza-se ponto.

Por exemplo, considere a seguinte matriz 3×3:

$$A = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 3 \end{pmatrix}$$

Para definir esta matriz no Scilab, pode-se usar o seguinte comando:

$$\mathrm{A} = [3, 2, 1; 2, 3, 2; 1, 2, 3]$$

2.2 Acessando e Modificando Elementos de uma Matriz

Após definida, pode-se acessar ou modificar elementos específicos de uma matriz. Por exemplo, considere a matriz A definida anteriormente. Para acessar um elemento particular,

utiliza-se a seguinte sintaxe:

$$a_{ij} = A(i, j)$$

onde i representa a linha e j a coluna desejada. Assim, o comando $A(1, 3)$ retorna o elemento que está na primeira linha e na terceira coluna, neste caso 1. De forma semelhante, $A(2, 2)$ retorna o valor 3 e assim sucessivamente.

Caso os valores de i e j estiverem fora do intervalo definido pela matriz, o programa irá retorna uma mensagem de *índice inválido*. Por exemplo, $A(1, 4)$ irá retornar esse erro pois a matriz A possui somente 3 colunas. Esta é, possivelmente, a causa mais comum de erros durante a implementação de métodos numéricos.

O comando $A(i, j)$ também pode ser utilizado para alterar ou definir um elemento específico da matriz A. Por exemplo, caso for necessário alterar o valor do elemento $A(2, 3)$ de 2 para 4, basta sobrescrever o valor:

$$A(2, 3) = 4$$

Os demais valores serão mantidos sem alterações.

Este comando também pode ser usado para definir um elemento que está fora das dimensões originais da matriz A. Por exemplo:

$$A(1, 5) = 1$$

Isto fará com que o elemento na primeira linha e na quinta coluna seja igual a 1. Automaticamente, a matriz A passará de uma matriz 3×3 para uma matriz 3×5. Para os demais elementos criados nesse processo, será atribuído o valor 0.

Dica de Sobrevivência 3: As dimensões de uma matriz podem ser facilmente observadas no navegador de variáveis, na coluna *Value*. No caso de matrizes, essa coluna apresenta as dimensões associadas. Um duplo clique no nome da variável irá abrir uma janela com os valores de cada elemento.

2.3 O Operador ‘ : ’

Um comando incrivelmente útil do Scilab é o ‘:’. De forma geral, ele pode ser interpretado como ***até***, e pode ser usado para acessar uma determinada sequência de valores de uma matriz. Considere novamente a matriz A definida anteriormente. Suponha que seja necessário

avaliar todos os elementos da coluna 2 da matriz e armazenar este vetor coluna resultante em outra variável $A2$. Isto pode ser feito através do seguinte comando:

$$A2 = A(1:3, 2)$$

Isto pode ser lido da seguinte forma: A variável $A2$ recebe os elementos da linha 1 até a linha 3 e da coluna 2 da matriz A. Em muitos casos, é desejável avaliar todas as linhas de uma coluna específica ou todas as colunas de uma linha específica. Neste caso, basta usar o comando ':' sem valor inicial ou final:

$$A2 = A(:, 2)$$

Isto pode ser lido da seguinte forma: A variável $A2$ recebe os elementos de todas as linhas e da coluna 2 da matriz A. Neste caso em particular, as duas formas irão gerar o mesmo resultado.

O comando ':' também pode ser usado para criar matrizes ou vetores com elementos igualmente espaçados. Por padrão, o passo utilizado para avaliar os elementos é igual a 1. Por exemplo:

$$B = 1:50$$

Este comando irá gerar um vetor com 50 componentes espaçados em 1 unidade, $B = (1, 2, 3, 4, 5, \ldots, 50)$

Também é possível definir um passo específico para separar os elementos, usando a sintaxe ***valor inicial : passo : valor final*** . Por exemplo:

$$B = 1:2:50$$

irá gerar um vetor com elementos espaçados em 2 unidades, desde o valor inicial 1 até o valor final 50, $B = (1, 3, 5, 7, \ldots 49)$. Observe que o valor final não está incluso pois com o passo definido o próximo valor seria 51, o que é maior que o valor final especificado. O passo especificado pode ser qualquer valor real, incluindo valores negativos.

Exercício 02: Utilizando o SciNotes, crie um vetor contendo 50 elementos, começando com o valor 200 e terminando com o valor 30. Faça com que todos os elementos esteja igualmente espaçados.

3 Loops e Estruturas Lógicas

Para a implementação de métodos numéricos, é comum a utilização de loops para repetir um determinado procedimento por um número definido de vezes ou até que um dado critério seja satisfeito. Os principais loops serão apresentados a seguir.

3.1 For

O comando *for* é utilizado para repetir uma dada operação por um número definido de vezes. Este é o comando básico para trabalhar com vetores e matrizes e permite fazer com que uma dada variável varie entre um valor inicial e um valor final com um passo fixo. Por padrão, o passo utilizado é igual a 1, porém qualquer valor real pode ser utilizado. Considere o seguinte exemplo:

```
1  //Valor inicial
2  valor_inicial = 1
3
4  //Passo desejado
5  passo = 1
6
7  //Valor final
8  valor_final = 10
9
10 //for variando i entre os valores inicial e final
11 for i = valor_inicial:passo:valor_final
12     //Definindo operações com a variável i
13     A(i) = 2*i+1
14 end
```

Neste caso, será criado um vetor A com 10 elementos, $A = (3, 5, 7, \ldots, 21)$. O mesmo resultado poderia ser conseguido utilizando o comando $A = 3 : 2 : 21$.

3.2 While

O comando *while* é um condicional que repete uma dada operação enquanto um critério de parada não for satisfeito. A principal utilização deste comando é na implementação de métodos iterativos, onde é avaliado um erro e o procedimento deve ser repetido até este erro ser inferior à precisão estabelecida. A utilização do while é ilustrada no exemplo a seguir.

```
1  //precisao desejada
2  prec = 1D-6 // = 10^(-6)
3
4  //inicializando o erro com um valor aleatório
5  erro = 1
6
7  //repetindo um procedimento enquanto o erro for
8  //maior que a precisão
9  i = 1 //índice do vetor A
10 A(1) = 10 //Posição 1 do vetor A
11 while erro>prec
12     A(i+1) = A(i)/2  //Criando uma posição i+1
13     erro = A(i+1)    //Avaliando a variável erro
14     i = i+1          //Avançando o contador i
15 end
```

3.3 If

Este comando é utilizado para executar uma dada ação se uma determinada condição for satisfeita, como mostrado no exemplo a seguir.

```
1  i = 5 //valor para comparação
2
3  for j = 1:10 //variando j de 1 até 10 com passo 1
4      if j<i then //se j for menor que i, então...
5          A(j) = 1
6      elseif j==i then //se j for igual a i, então...
7          A(j) = 2
8      else //caso as outas condições não forem satisfeitas
9          A(j) = 3
10     end
11 end
```

4 Criando e Utilizando Funções

Do ponto de vista de programação, o termo ***função*** se refere a uma rotina qualquer que relaciona uma ***entrada*** com uma ***saída***. Este conceito é muito mais amplo do que o conceito de função matemática. Por exemplo, pode-se utilizar uma função para plotar um conjunto de dados, ou seja, uma função que relacione dois vetores (entradas) com um gráfico (saída). Não existe restrição no formato ou na quantidade de entradas ou saídas, sendo isso definido pelo usuário.

Uma função matemática $y = f(x)$ também representa uma relação entre uma entrada x e uma saída y, portanto pode-se criar um função no Scilab equivalente. Porém, também é possível criar funções que relacionem quantidades com dimensões distintas, por exemplo relacionar 2 vetores com um escalar.

A utilização de funções facilita muito a implementação de códigos, especialmente se estes forem mais complexos. De modo geral, as funções são definidas para repetir um dado procedimento sem que seja necessário reescrever todas as etapas a cada vez.

A sintaxe básica da definição de uma função contém 3 elementos:

1. Parâmetros de entrada;
2. Parâmetros de saída;
3. Nome da função.

Todos estes elementos são definidos pelo usuário. Para mostrar a utilização de uma função, considere um exemplo simples onde se deseja definir uma função que relacione um valor de entrada com um valor de saída, seguindo a relação:

$$y = x^2 + 3x + e^x$$

onde x é o parâmetro de entrada e y é a saída, ou seja, a função deve retornar o valor de y quando um valor de x for especificado.

O primeiro ponto a ser identificado são quais são as entradas e as saídas e quais os formatos destes termos. Neste caso temos:

- Parâmetros de entrada: 1 parâmetro, escalar;
- Parâmetros de saída: 1 parâmetro, escalar.

Para definir uma função no Scilab, a sintaxe geral é a seguinte:

```
function [saida1, saida2, ...]=nome_da_funcao(entrada1, entrada2, ...)
    //relação entre as saídas e as entradas
    saida1 = ...
    saida2 = ...
    ...
endfunction
```

O nome dado para as variáveis de entrada e saída é completamente aleatório e ***não*** está associado com as demais variáveis e parâmetros definidos no código. Dentro do ambiente de definição das funções, as variáveis de entrada e saída são tratadas como locais. A função $y(x)$ anterior pode ser definida da seguinte forma:

```
function saida1 = fun(entrada1)
    saida1 = entrada1**2 + 3*entrada1 + exp(entrada1)
    // ** representa potência
    //exp() representa a função exponencial
endfunction
```

Com isso, cria-se uma função chamada *fun* que relaciona uma entrada (escalar) com uma saída (escalar). Para acessar esta função, é preciso definir o valor desejado para a entrada e atribuir este valor na função. Por exemplo, considere que seja necessário avaliar o valor de y para $x = 2.76$. Isto pode ser facilmente realizado da seguinte forma:

```
x = 2.76
y = fun(x)
```

Assim, a variável y passa a receber o valor $2.76^2 + 3(2.76) + e^{2.76}$.

Considere agora que seja necessário criar um função relacionando um vetor com três componentes como entrada $x = (x_1, x_2, x_3)$ com um vetor com duas componentes como saídas $y = (y_1, y_2)$, da seguinte forma:

$$y_1 = x_1 x_2 + x_3^{x_1} \tag{4.1}$$

$$y_2 = x_1 + x_2 + x_3 \tag{4.2}$$

Neste caso, também deve-se relacionar uma entrada com uma saída, porém ambas são vetores e não escalares. Para isso, basta especificar qual componente dos vetores deve ser acessada, como mostrado a seguir:

```
function saida1 = fun2(entrada1)
    saida1(1) = entrada1(1)*entrada1(2) + entrada1(3)**entrada1(1)
    saida1(2) = entrada1(1) + entrada1(2) + entrada1(3)
endfunction
```

É importante observar que para avaliar esta função, a entrada especificar deve ter ***exatamente*** as mesmas dimensões da entrada definida na função. Por exemplo, para avaliar o vetor y com base em uma entrada $x = (1.25, 2, 3.5)$, as variáveis devem ser definidas como:

```
x = [1.25, 2, 3.5]
y = fun2(x)
```

Com isso, cria-se um vetor y com duas posições, de acordo com a função definida.

Exemplo: Crie um função que permita calcular a pressão de um gás em função da temperatura e do volume específico utilizando a equação de Van der Waals:

$$P = \frac{RT}{V - b} - \frac{a}{V^2}$$

onde as constantes a e b são definidas em função das propriedades críticas do fluido avaliado:

$$a = \frac{27(RT_c)^2}{64P_c} \qquad b = \frac{RT_c}{8P_c}$$

Faça com que a função possua 4 parâmetros de entrada: a temperatura (K), o volume (cm^3/mol) e as propriedades críticas.

Utilize a função para avaliar a pressão em um sistema contendo CO_2 a 500K e com volume específico de 25000 cm^3/mol. Para o dióxido de carbono, $T_c = 304.2\,K$ e $P_c = 73.85\,bar$.

```
function saida = VdW(ent1,ent2,ent3,ent4)
//ent1 = temperatura, ent2 = volume, ent3 = Pc, ent4 = Tc
    //Associando as entradas com parâmetros
    T = ent1
    V = ent2
    Pc = ent3
    Tc = ent4
    //Definindo a constante R
    R = 83.14 //cm^3 bar/mol K
    //Calculando os parâmetros a e b
    a = (27*(R*Tc)**2)/(64*Pc) //constante de VdW
    b = (R*Tc)/(8*Pc)
    //Calculando a pressão com base nas entradas
    saida = (R*T)/(V-b) - a/V**2
endfunction

//Calculando a pressão para o CO2
PCO2 = VdW(500,25000,73.8,304.2)
```

4.1 Funções Definidas no Scilab

Existem diversas funções pré-definidas no Scilab, por exemplo:

- ***plot:*** esta função é utilizada para fazer gráficos 2D, plotando um vetor em função de outro. A sintaxe é muito simples, basta definir $plot(vec1, vec2)$. Como resultado, será gerado o gráfico relacionando os pontos do vetor $vec1$ no eixo x e $vec2$ no eixo y. Esta função só funciona se $vec1$ e $vec2$ possuírem exatamente o mesmo número e elementos.

- ***numderivative:*** esta função é utilizada para aproximar a derivada de uma dada função em um ponto. Por exemplo, considere a seguinte função:

$$y = x^2 \cos(x) e^x$$

Suponha que seja necessário avaliar a derivada desta função em dez pontos igualmente espaçados entre 0 e π. Isto pode ser feito da seguinte forma:

```
//Limpando a memória e o console
clear
clc

//Definindo a função
function saida1 = f(entrada1)
    saida1 = (entrada1**2)*cos(entrada1)*exp(entrada1)
endfunction

//Criando o vetor de pontos onde será avaliada a derivada
N = 10 //Número de pontos - número de divisões será N-1 = 9
x = [0:%pi/(N-1):%pi] //%pi representa a constante pi

//Avaliando a derivada em cada um dos pontos
for i = 1:N
y(i) = numderivative(f,x(i))
end
```

5 Exercícios Propostos

1) Considere o seguinte sistema linear de 100 equações.

$$T_1 = 0$$

$$T_{i-1} - 4T_i + T_{i+1} = 0 \qquad i = 2, \ldots, 99$$

$$T_{100} = 1$$

Escreva este sistema na forma matricial, $\mathbf{A}\vec{T} = \vec{b}$, onde $\mathbf{A}$ é uma matriz 100 x 100, $\vec{T} = (T_1, T_2, T_3, \ldots, t_{100})^T$ e $\vec{b} = (0, 0, 0, \ldots, 1)^T$. Defina a matriz $\mathbf{A}$ e o vetor $\vec{b}$ no Scilab. Resolva o sistema utilizando o algoritmo TDMA.

2) Crie um função que permita calcular o volume específico de um gás em função da temperatura e da pressão utilizando a equação de Van der Waals. Utilize o método de Newton para resolver a equação não-linear.

Dica: Como chute inicial, calcule o volume específico considerando que o gás possua um comportamento ideal;

Dica 2: Dentro de uma função, pode-se definir outras funções.